MILWAUKEE AREA ATARI USERS CLUB

MILATARI NEWSLETTER

Volume 3 Number 5

April 1984

Price $1.00

** NEXT MEETING **

SATURDAY, April 21st

ARMBRUSTER SCHOOL - GREENDALE

ADGENDA:

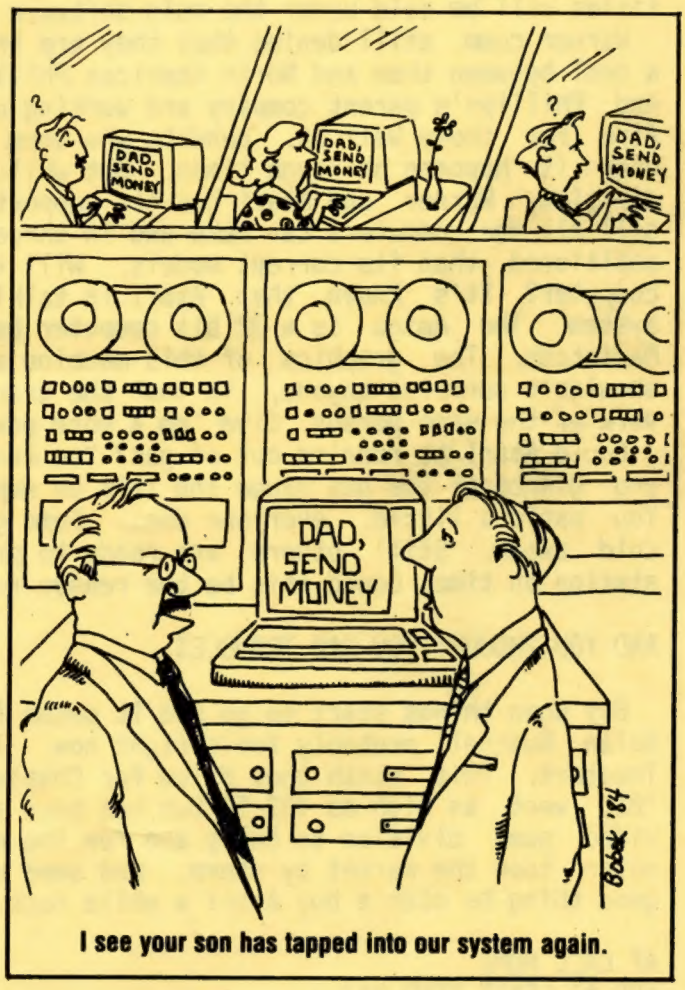
2:00PM Basic Class
Assembler Class
Libraries open*
Kids Corner open

3:15PM Business Meeting

4:00PM Demonstrations

* * * * * * * * * * *

*Libraries will be closed during the business meeting and demonstrations



I see your son has tapped into our system again.

PRESIDENT'S RAM

by Gary Nolan

DROVE MY CHEVY TO THE LEVEE, BUT THE CHEVY RAN DRY

Several articles in the last couple of weeks have detailed the blood letting at ATARI. The latest moves by Gentleman Jim have some people shook-up. One of those people is none other than Chris Crawford, who was told by the big "A" that his services were no longer needed. That's right, CHRIS WAS CANNED! He along with about 100 others got their pink slips the other day. As a spokesman for Atari (I think it was a Mr. Fottin Mouth) said, "It's a personal tragedy for those who are affected..." Atari has also said that they are curtailing long range research and research into software that doesn't appeal to the broadest audience. They will also concentrate future research on products and not on "blue-sky projects".

Atari also laid off all employees involved in the manufacture of APX software. While some titles will be sold under the main software line, the fate of the APX division is still unknown.

Warner comm. still denies that they are trying to peddle the Atari division, but the rumors of a deal between them and North American Phillips is still strong on Wall St. The fact that Atari and Phillips's parent company are working on a joint video game project only adds fuel to the fire for those Wall St. pundits who seem to delight in causing the early demise of a company when it happens on rough times. But while there are those who have written off the computer division, Warner and Atari has its supporters. One goes so far as to predict that Atari will sell all the computers can make and in three to nine months will unveil a product that is better positioned than its current models. Will that new machine be a low cost copy of the new Amiga computer? It's known that Atari is talking with Amiga about a licensing agreement for its system. The Amiga is a 32 bit computer based on the same 68000 chip that's used in the new Macintosh. The graphics of this machine are supposed to be nothing short of fantastic. That shouldn't surprise anyone, in that the graphic system of the both the Amiga and Atari were the work of the same person. Give 'em a more powerful CPU and he'll give you a better display.

Could Atari be running out of gas? You know that feeling you get when you're driving along and you glance at the gas gauge and it's on empty and you don't know where the next gas station is? You panic a little, everyone does. Some get a knot in their stomachs and some break out in a cold sweat. Still others are ready to promise the good Lord anything to get them to a gas station on time! Could this be the reason for "hacking" going on at the big "A"?????

AND YOU THOUGHT YOU HAD TROUBLES

Boy when things start to go bad it seems like there's no tomorrow. That's how our old friend Nolan Bushnell probably feels right now. In Febuary he stepped down as chairman of Pizza Time Theaters. This month they filed for Chapter 11. Its stock sold for $15 a share at release in '81, went as high as $32.50 but has been sliding ever since. Sente Technologies has sold its video game division to Bally and for the most part it's having troubles of its own. Androbot never took the market by storm, and some of its other start-ups are sputtering. Maybe it's a good thing he didn't buy Atari a while back.

AF CALL HOME
(OR AT LEAST SEND MAIL)

Has anybody heard from Austin Franklin lately? Several people from our group (me included) have been trying to reach AF by phone for several weeks. I sent the 80 col. board back for upgrading on the 18th of Jan. and have heard nothing since. Calling at various times of the day and night have resulted in no answer. Just when I've found a good use for the board, with the ATR8000, it's nowhere to be found. It's a shame too, because once I got a board that worked, I really got to like it. I still think that it offers more for the money than the Bit3. Now if only I could try it with some good 80 col. CP/M software and the ATR.....

SOME OODS AND ENDS
(THE ENDS OF WHAT?)

From our own Vern Socks comes a mailing list and label printing program. The ML program will handle 100 records and will be sold through the disk library for five dollars. On the back side will be a label printing program.

Blank disks should be available at the meeting for $19 a pack of ten. These are single sided, double density, soft sector disks. They usually sell pretty quick, so get there early.

MAYBE YOU DIDN'T LISTEN RIGHT
(OR MAYBE I DIDN'T WRITE IT RIGHT...NAAA)

In last months column I talked about Atari's support (or lack of) for its user groups. What it really was, was kind of worrying out loud. With the axe being swung wide and hard at "non-essential programs", I was hoping that the User Group Support Div. would not be cut, and maybe even enlarged. Most groups around the country do more PR for Atari than the big "A" really knows about.

OK! So coffee mug and pens may not be a big deal. But bringing in the area rep and flying in the factory UG man for a group of 50 (at most) is. That's what Kaypro did last month. I don't expect to see anyone from Atari at our meetings, that's not why I brought the whole thing up. It was kind of hoping out loud. But if it works maybe I'll hope out loud for a great 32 bit color graphics machine with built in 3 1/2" drives that can be expanded from here to never-never land.

BIG BITS

THERE WILL BE NO WORKSHOP THIS MONTH

In order to retain my sanity and control my ulcer, I'm going to ask for some help from some Milatari members. I need someone to help me set-up and run the workshops. What this will involve is to take names of people willing to demo hardware or software, to help run the sessions (easy, introduce the people and let them talk) and to help plan upcomming workshops. Not really much work for somebody out there willing to help. Call me or see me at the meeting.

LITTLE BITS

Have an XL computer that showing a little more interference than you think it should? Try plugging the end of the RF cord that has that round thing on it in to the switch box. Some members have found that it gives a clearer picture.

Has anybody noticed that there are two different keyboards on the 800XL's. I noticed this when I was at American. On one keyboard the keys had a slight space between the keys, and you could see the circuit board the keys are mounted on. On the one that I have the keys are close together and there is a black plastic "mask" under them. The feel is different between them also. I still like the one that's on mine.

Is anyone having trouble getting the Amodem programs to work with the new XL's? See a reprint from one of the other user groups elsewhere in this newsletter.

How many of you members go to Arlington Park during the summer to "play the ponies"? Well there's a program to help turn the odds in your favor. At least that's what they claim. It comes on cassette for $30 and is from 3G Company Inc. If anyone is interested see me at the meeting.

One of the new computer magazines to hit the stands claims to serve those systems that will me the prime home computers for the 80's. They are Apple, C-64, TI-99/4A and I think the Adam. No mention was made of the Atari. Even HI-RES which started out as an Atari only mag has hedged its bets by including coverage of the 64.

SINCE YOU KIDS CAN'T GET ALONG,
WE'LL JUST HAVE TO SEPARATE YOU

And now it's time for our second installment of Kids In BBSland. Our story begins with swearing and name calling among the children, which leads to the shut down of the BBS. This in turn angers some of the users of the BBS. And finally leads to the installation of the password system, with the threat of legal recourse by the operators of BBSville, against any naughty children.

Sound DUMB???

So do the things that have been happening on the MILATARI BBS.

Anyone who has been on the board since April 1st knows that some changes have been made and more are coming. To those who have been put out by this, I'm sorry. But after we're done we should have better control over the board, and that can only make calling it a more enjoyable time. Thanks for your patience.

TILL THEN

I know I'll hate myself in the morning for forgetting something, but the deadline draws near and Dave is beating on my door.
SEE YOU ON THE 17th.....
HAPPY EASTER EVERYBODY!

* *

MODIFYING AMODEM FOR USE ON XL COMPUTERS

From KEEPING PACE Pittsburgh, PA.

This information will allow those of you who have XL computers to use AMODEM series of telecommucation programs. Please note: These modifications will make the AMODEM program you modify unusable except on an XL computer. I corrected an illegal jump with another illegal jump.

The modifications are simple and will allow you to use the XMODEM protocol when uploading or downloading. Load your AMODEM program and list it. You now will have to look for two strings, SEND$ and REC$. When you find where these strings are being assigned to a bunch of inverse graphics and letters, stop. In SEND$, you will want to change the inverse $ and inverse v (the v is lower case). They will appear in that order. To change SEND$, simply position the cursor over the $ and type an inverse zero (0). Next type an inverse lower case r. then press RETURN.

Move on to REC$ and again position the cursor over the inverse $ and type an inverse zero (0) then an inverse lower case r. Again press the RETURN key.

To be sure of your work, you are changing bytes number 59 and 60 in SEND$ and bytes number 70 and 71 in REC$.

Now save your program. You will be able to use the AMODEM program with your XL computer. Just remember, the modified program will not work on the older 400/800 computers. The un-modified program will not worl on the new XL's, so keep a version handy for each type of computer, the simply choose the program that suits the computer.

As a final note, I saved my modified program as AMPLUS.XL so I would know that it was for my XL.

Rich Strecker - 71425,1637

SEEK and FIND

ASCII: American Standard Code for Information Interchange. A coding scheme wherein letters, numberd and special symbols are represented as unigue 7 bit values, allowing for standardization between data communication devices.

ASYNCHRONOUS communication. A serial stream of data sent as generated. Characters are deliminated by start and stop bits whose function is to synchronize character bit timing.

CRC: A method of error detection using Cyclic Redundancy Check characters(s). A CRC character is generated at the transmitting terminal based on the contents of the message transmitted. A similar CRC generation is performed by the receiving terminal. If the two characters match, the message was properly received.

EBCDIC: Extended binary coded dicimal interchange code. A coding scheme wherein letters, numbers and special symbols are represented as unique 8 bit values, allowing for standization between data communications devices. It was popularized by IBM.

EPROM: Erasable programmable read only memory. Ultraviolet light erases programming.

HERTZ: A unit of frequency (Hz) equal to one cycle per second. The AC line frequency in the U.S. is 60Hz; the voltage changes polarity 120 times per second.

MODEM (or data set): A communications device whose function is to convert digital signals to analog and vise versa. Modem derives from the term modulator/demodulator.

NANOSECOND: One-billionth of a second - faster than the human eye can blink. One nanosecond is to one second as one second is to 32 years Computers work at speeds measured in nanoseconds.

PERIPHERAL: A noncomputing input or output device, such as a printer or hard disk.

PROM: Programmable read-only memory Requires special devices to alter programming.

THE 10 HIDDEN WORDS ARE

ASCII	ASYNCHRONOUS
EBCDIC	EPROM
PROM	CRC
NANOSECOND	PERIPHERAL
HERTZ	MODEM

N	L	O	W	G	Y	Q	C	A	A	V	G	A	L
R	A	H	O	V	L	N	J	R	U	J	S	B	V
U	R	L	V	N	I	F	S	S	C	Y	W	S	D
Q	E	B	O	C	L	I	L	M	N	F	A	O	P
Y	H	J	J	V	B	F	O	C	D	G	W	E	D
S	P	E	Q	L	M	R	H	E	B	C	D	I	C
U	I	B	I	O	P	R	Z	U	R	W	X	N	R
A	R	S	R	E	O	K	A	T	T	T	M	Z	O
S	E	P	G	N	O	M	G	S	R	E	R	E	A
C	P	O	O	Y	S	Y	V	Y	D	E	F	F	A
I	Z	U	Z	K	C	F	P	O	P	M	H	T	C
I	S	T	R	H	P	P	M	O	S	S	Y	Z	R
S	I	H	D	N	O	C	E	S	O	N	A	N	C
B	S	J	Y	B	X	X	E	A	K	R	H	S	I

DEMOPAC #8

SOFTWARE AND HARDWARE TIMERS

Examples and discussions of using Software and Hardware Timers using BASIC with machine language routines

1) Software duration timers
2) Software background timer
3) Hardware timer

Information provided by:
ATARI Inc.
CONSUMER PRODUCT SERVICE
PRODUCT SUPPORT GROUP
1312 Crossman Avenue
Sunnyvale, CA 94086

- -

SOFTWARE TIMERS
JC 5/10/83

The ATARI operating system maintains five software timers. These timers are actually memory locations which are update (decremented) during the VBLANK interrupt process.

'VBLANK' refers to the interrupt which occurs each 1/60th second at the end of a complete video scan, while the video beam is shut off and returned to the upper left corner of the screen. During this period the operating system performs various system functions in two stages. Stage one VBLANK processes are always performed every VBLANK or 1/60th second. Stage two VBLANK functions may be skipped if time critical operations must be performed, as indicated by a non zero value in the 'CRITIC' flag at decimal location 66.

Timer number one is used by the operating system to time I/O operations and is updated during each stage one VBLANK. Timer number one should not be used by an application program unless serial I/O is suspended during the period it is in use.

Timers two through five are not used by the operating system and are updated every stage two VBLANK. Since stage two VBLANK is skipped during time critical I/O operations, timers two through five may not always be update every 1/60th second. This can be seen by running the background timer demonstration program and then performing an I/O operation such as accessing the disk to read or write a file. If it is essential not to miss a count while using a timer during critical I/O operations, the system clock which is updated during stage one VBLANK may be used.

There are two methods used by the timers to indicate that the countdown has been completed. Timers one and two call a subroutine. An applications program may place the address of a user subroutine into special memory locations reserved for this purpose. When timer one or two times out (decrements to zero), a 'JSR' to the usersubroutine takes place. The user subroutine should end with an RTS (return) instruction. Timers two through five use special locations in memory as 'flags'. It is necessary to set the 'flag' to a non-zero value before starting the timer and then check the 'flag' location for a zero value, which indicates that time out has occured.

Since the timers are incremented in 1/60th second intervals, the minimum time value that can be set is 1/60th second (when a value of 1 is used), and the maximum time value is 18.20 minutes (when a value of 65,535 or hex $FFFF, is used).

TO SET and START THE TIMERS USE THE FOLLOWING PROCEDURE:

1. For timer 1 or 2, store the address of the routine to be executed when time out occurs, in the vector locations specified in the Software Timer Address Table. Store the Least Significant Byte (LSB) followed by Most Significant Byte (MSB). Make sure the code to executed at this address has been loaded into memory. For timers 3,4, or 5 set the flag location specified in the table, to the value '$FF' (this location becomes '0' when time out occurs).

2. Load the accumulator with the number (1-5), of the software timer.

3. Set the timer duration by loading the 'Y' register with the LSB, and the 'X' register with the MSB of the number of 1/60th second intervals to count. A value of '0' in the 'X' register and '60' in the 'Y' register would indicate a 1 second interval.

4. To begin timer execution do a JSR (Jump Subroutine) to the 'SETBYTE' routine at hex location $E45C.

5. If timer 3,4, or 5 is used it is necessary to check its flag location for a zero value, to determine if time out has occured. Since timers 1 and 2 are vectored to the address in step 1, when time out occurs, execution will automatically begin at the specified location. This makes it possible to create a program which re-initializes and re-starts the timers just before returning, giving a background effect. This is the method employed by the 'Background Software Timer' demonstration program which plays a song in the background while other activities may continue in the foreground.

The following chart lists the locations of the five software timers and their respective flags and jump vectors. All address are in hexadecimal.

SOFTWARE TIMER ADDRESS TABLE

Timer Number	Timer Name	Hex Addr.	Flag/ Vector	Hex Addr.	Use
1	CDTMV1	$0218	CDTMA1	$0226	2-byte vector
2	CDTMV2	$021A	CDTMA2	$0228	2-byte vector
3	CDTMV3	$021C	CDTMF3	$022A	1-byte flag
4	CDTMV4	$021E	CDTMF4	$022C	1-byte flag
5	CDTMV5	$0220	CDTMF5	$022F	1-byte flag

SOFTWARE TIMER DEMO PROGRAM

SOFTIME.BAS is a program which demonstrates the use of the software timers (2-5), which set a flag to indicate that the timer has competed its countdown to zero. When the timer has competed its assigned task, a location in memory (see timer address table), is set to zero.

The following BASIC program uses an assembly language routine, SOFTIME.ASM, to set and start software timer number four. The assembly language routine then continuously checks the associated timer flag until it becomes zero, at which time a return instruction is executed, and program control returns to the BASIC program.

Using the software timers in this way is an extremely simple process. The assembly language portion of this routine uses only ten instructions and takes up a compact 16 bytes. In spite of its small size, because the time duration is passed from BASIC, it is possible to use this routine for a variety of purposes in a BASIC program. The routine can replace many FOR/NEXT loops used for timing durations, when more accurate timing is important.

In the SOFTIME.BAS example, the timer routine is used three times with varying values to time different functions. In line 350 the user is prompted ti enter the timer duration in Jiffies (1/60th second), which is stored as the numeric variable D. This value is then used in line 480 to time the delay between printing characters on the screen. The value of D can vary between 1 (1/60th second) and 65535 (18.20) minutes). Before the character is printed, the timer is used to time a sound of 1/60th second duration in line 450. After the screen border has been filled with characters, the timer is used for a third function in line 500 where a two second delay is timed prior to clearing the screen and repeating the entire process.

```
100 REM ****************************
110 REM * SOFTIME.BAS              *
120 REM *                         *
130 REM * John Clark              *
140 REM * 04/21/83                *
150 REM * call assembly language  *
160 REM * routine to use software *
170 REM * timer #4 to time BASIC  *
180 REM * routines                *
190 REM ****************************
200 REM
210 REM ****************************
220 REM SET UP ARRAY FOR SCREEN POSITIONING
230 DIM X(55),Y(55)
240 FOR I=0 TO 55
250 IF I<20 THEN X(I)=I:Y(I)=0
260 IF I>19 AND I<29 THEN X(I)=19:Y(I)=I-19
270 IF I>28 AND I<48 THEN X(I)=-1*(I-29)+18:Y(I)=9
280 IF I>47 THEN X(I)=0:Y(I)=-1*(I-48)+8
290 NEXT I
300 REM ****************************
310 GOSUB 610:REM POKE ASSEMBLY LANGUAGE ROUTINE INTO MEMORY
320 REM
330 REM ****************************
340 REM CALL GR.2 AND SET COLORS
350 GRAPHICS 2
360 POKE 712,190:REM BACKGROUND COLOR
370 POKE 710,246:REM LETTER COLOR
380 POSITION 2,5:PRINT #6;"software timers"
390 ? "INPUT TIMER DURATION (JIFFIES)";
400 INPUT D
410 REM ****************************
420 REM LOOP TO PRINT TIMED DISPLAY
430 FOR I=0 TO 55
440 SOUND 0,230,10,8
450 A=USR(1536,1):REM TIME SOUND DURATION 1/60TH SECOND
460 SOUND 0,0,0,0
470 POSITION X(I),Y(I):? #6;CHR$(176):REM PRINT BLUE ZERO CHARACTER
480 A=USR(1536,D):REM TIME DURATION OF EACH SCREEN CHARACTER DISPLAYED
490 NEXT I
500 A=USR(1536,120):REM TWO SECOND DELAY AND THEN REPEAT PROMPT
510 GOTO 350
520 REM
530 REM
540 REM ****************************
550 REM POKE ASSEMBLY LANGUAGE
560 REM TIMER PROGRAM
570 REM INTO MEMORY
580 REM BEGINNING AT LOCATION 1536
590 REM ****************************
600 REM
610 FOR I=0 TO 20
620 READ J:POKE 1536+I,J
630 NEXT I
640 RETURN
650 DATA 104,104,170,104,168,169,255,141,44,2
660 DATA 169,4,32,92,228,173,44,2,208,251,96
```

```
05 ;*********************************
10 ;* SOFTWARE DURATION TIMERS      *
15 ;*                               *
20 ;*   John Clark 5/10/83          *
25 ;*                               *
30 ;* timer basic operations       *
35 ;* with assembly language       *
40 ;* routines.                    *
45 ;*                               *
50 ;* Called from BASIC with       *
55 ;* 'A=USR(1536,Duration)'       *
60 ;* where duration is number     *
65 ;* of 1/60 second units         *
70 ;*                               *
75 ;*********************************
80 ;
85 ;
90 CDTMF4 = $022C    ;countdown timer #4 flag
95 SETVBV = $E45C    ;set timer routine
0100 ;
0105  *=$600
0110 ;
0115  PLA            ;throw away number of arguments
0120  PLA            ;MSB of timer duration
0125  TAX            ;store in X register
0130  PLA            ;get LSB of timer duration
0135  TAY            ;store in Y register
0140  LDA #$FF       ;value to initialize timer #4 flag
0145  STA CDTMF4     ;store in timer #4 flah
0150  LDA #4         ;indicate timer #4 will be used
0155  JSR SETVBV     ;set timer and start timer
0156 LOOP LDA CDTMF4 ;check timer #4 flag
0158  BNE LOOP       ;if not '0' then check again
0160  RTS            ;return to BASIC
```

BACKGROUND SOFTWARE TIMER DEMO PROGRAM

The Background Timer demonstration uses software timer #2 to play a background tune, while other activity, such as running or editing a BASIC program, can take place in the foreground.

The BASIC portion of this program simply pokes the assembly language routine into memory and calls it with a USR statement. Once the program is initiated, it will continue running until system reset is pressed.

This program takes advantage of the fact that when software timer number has finished its countdown, a jump subroutine takes place to designated user routine. The address of this routine is placed in the timer number two vector location at hex $0228 and $0229 (LSB,MSB). The background effect is achieved by having the user routine re-o and start timer number two after playing a note from the note table stored in memory. Each time the timer counts down to zero, a jump subroutine occurs to the user defined interrupt service routine, which plays a note and starts the timer again. An offset to determine which note will be played from the note table stored in location $0676. This offset is increased to point to successive notes each time the routine is called, and is reset to zero, to begin again, after all thirty notes have been played, Software timer number three is used to time a 1/30th second duration for each note.

Since timer number two is updated during each stage two VBLANK, any I/O activity will interface with the background operation (stage two VBLANK operations are skipped when critical I/O is taking place). Initiate the program and then do a disk file access. The background song will stop and then resume when I/O has been completed. If it is important to keep timing while I/O is taking place, the system clock may be used.

```
0100 REM ************************
0110 REM *** BACKGROUND TIMER ***
0120 REM ***   John Clark     ***
0130 REM ***    04/18/83      ***
0140 REM ************************
0150 REM
0160 REM ******************************************
0170 REM *** POKE MACHINE LANGUAGE TIMER ROUTINE ***
0180 REM *** INTO PAGE SIX MEMORY                ***
0190 REM *** AND CALL WITH USR FUNCTION          ***
0200 REM ******************************************
0210 REM
0220 REM
0230 FOR I=0 TO 118
0240 READ J:POKE 1536+I,J
0250 NEXT I
0260 A=USR(1536)
0270 END
0280 REM
0290 REM
0300 DATA 104,32,66,6,162,0,142,118,6,96,169,0,141,8,210,169,3,141,15,
         210,174,118,6,189,86,6,141,0,210,232
0310 DATA 224,32,208,2,162,0,142,118,6,169,175,141,1,210,160,
         2,162,0,169,255,141,42,2,169,3,32,92,228,173,42
0320 DATA 2,208,251,141,1,210,169,10,141,40,2,169,6,141,41,2,
         169,2,160,20,162,0,32,92,228,96,96,108,121,108
0330 DATA 96,96,96,0,108,108,108,0,96,81,81,0,96,108,121,108,
         96,96,96,0,108,108,96,108,121,0,0,0,0

10 ;****************************
20 ;BACKGROUND SOFTWARE TIMER
30 ;****************************
40 ;
50 ;John Clark 04/18/83
60 ;use software timer #2 interrupt
70 ;to play background tune
80 ;use software timer #3 to count duration of note
82 ;called from BASIC with
83 ;'A=USR(1536)'
90 ;
91 ;****************************
92 ;
0100 CDTMA2 = $228  ;vector for timer interrupt
0110 SETVBV = $E45C ;set timer routine
0120 AUDCTL= $D208  ;audio control
0130 SKCTL = $D20F  ;serial port control
0140 AUDF1 = $D200  ;audio freq 1
0150 AUDC1 = $D201  ;audio channel 1 control
0170 CDTMF3 = $22A  ;timer 3 flag
0180 ;
0190  *=$600
0200 ;INITIALIZE TIMER #2 AND RETURN
0210 ;
0220 ;add PLA here if calling from BASIC
0230 ;
```

```
0240  JSR INIT     ;initialize timer #2
0250  LDX #0       ;initial offset value for note table
0260  STX POINT    ;save offset to note table in memory
0270  RTS          ;return from initializing timer #2
0280 ;*******************************
0290 ;TIMER INTERRUPT SERVICE ROUTINE
0300 ;
0310 TIME LDA #0     ;initialize audio control
0320  STA AUDCTL
0330  LDA #3        ;initialize POKEY to use audio registers
0340  STA SKCTL
0350  LDX POINT     ;get offset to note table from memory
0360  LDA NOTES,X   ;get a note from note table
0370  STA AUDF1     ;place note in audio freq register
0380  INX           ;point to next note in table
0390  CPX #32       ;is it last note in table?
0400  BNE STORE     ;no, store note in memory location POINT
0410  LDX #0        ;if done poin
0420 STORE STX POINT ;store next offset in memory
0430  LDA #$AF      ;use pure note full volume
0440  STA AUDC1     ;place in audio control register
0490  LDY #2        ;1/30 second duration
0520 COUNT LDX #0    ;MSB of note duration
0540  LDA #$FF      ;init timer 3 flag
0550  STA CDTMF3    ;to a non zero value
0551  LDA #3        ;indicate set timer #3
0560  JSR SETVBV     ;set and start timer #3
0570 ;***************************
0580 ;loop to time duration
0590 ;***************************
0600 LOOP LDA CDTMF3 ;check timer flag
0610  BNE LOOP      ;check until  0
0620  STA AUDC1     ;found 0 in a register, turn off sound
0630 ;
0640 ;*********************************
0650 ;TIMER #2 INTERRUPT INITIALIZATION ROUTINE
0660 INIT LDA #TIME&255   ;LSB of background routine
0670  STA CDTMA2          ;set vector for timer #2 interrupt
0680  LDA #TIME/256       ;MSB of background routine
0690  STA CDTMA2+1        ;set vector for timer #2 interrupt
0700  LDA #2             ;indicate software timer #2 is to be used
0710  LDY #20            ;LSB of countdawn valur for timer #2
0720  LDX #0             ;initiate timer #2
0730  JSR SETVBV         ;initiate timer #2
0740  RTS
0750 ;
0760 ;define notes for background song
0770 NOTES .BYTE $60,$6C,$79,$6C,$60,$60,$60
0780  .BYTE $00,$6C,$6C,$6C,$00,$60,$51,$51,$00
0781  .BYTE $60,$6C,$79,$6C,$60,$60,$60
0782  .BYTE $00,$6C,$6C,$60,$6C,$79,$00,$00,$00
0790 POINT .BYTE 0
0800  *=$02E0
0810   .WORD $600
```

HARDWARE TIMERS
JC 5/10/83

The POKEY chip contains four countdown timers which are commonly used for sound generation. Three of these hardware timers also have interrupt vectors which can point to a user defined routine. Because of the high clock speeds used with these hardware timers, the software timer limitation of 1/60th second can be overcome. It should be noted that hardware timers number two and three are used during serial I/O, for baud rate generation and cannot be altered.

The POKEY timers by default operate with a clock rate of 64 Khz (64000 cycles per second), but it is possible, by setting the correct bits in the AUDCTL register, to change this rate to 15 Khz (15000 cycles per second) or 1.79 Mhz (1,790,000 cycles per second). The number of cycles to count before generating an interrupt is stored in the audio frequency registers, AUDF1, AUDF2, or AUDF4. It is possible to merge two audio frequency registers by setting a bit in the AUDCTL register and use two bytes to store the value representing the number of cycles to count. The hardware timer demonstration program used this technique. This makes the maximum duration that can be timed equal to 4.369 seconds and the theoretical minimum duration that can be timed equal to 1.79 millionths of a second. When using the hardware timers to time extremely short intervals, it is necessary to turn off DMA and Vertical Blank Interrupts (VBI) by storing a 0 in DMACTL at location $D400, and NMIEN at location $D40E. Experiment with your particular application to determine if the timing rate generated is in conflict with the screen display.

TO SET AND START THE HARDWARE TIMERS USE THE FOLLOWING PROCEDURES:

1. Set the AUDCTL ($D208) register to choose the desired clock speed. The default clock speed is 64 Khz. To enable the 15 Khz clock for all channels, set bit 0. To enable the 1.79 Mhz clock for channel number 1, set bit 6, and for channel number 2 set bit 5. AUDCTL can also be used to join channels 1 and 2, or channel 3 and 4, to allow the timing duration to be defined by sixteen bits. This provides a range of from 1 to 65,535 clock cycles.

2. Set the volume level for the channel(s) to be used as timers, to 0. by storing a 0 in the AUDC1 ($D201), AUDC2 ($D203), or AUDC4 ($D207) register.

3. Choose the number of clock cycles to count and store this value in the frequency register for the appropriate channel, AUDF1 (ZD200), AUDF2 ($D202), or AUDF4 ($D206). If two channels have been joined to allow a sixteen bit value to be specified for the number of clock cycles, enter the Least Significant Byte (SLB) in the lower numbered frequency register of the pair, and enter the Most Significant Byte (MSB) in the higher numbered frequency register.

4. Set up the routine to process the interrupt which will occur when the timer has completed its count. It is possible to have this routine re-initialize and re-start the hardware timer, so that the interrupt routine will be performed continuously in the background (see example program).

5. Set the timer interrupt vector to point to the routine created in step 4. Place the address of this routine (LSB,MSB) in the vector register associated with the timer selected, (VTIMR1-$0210, VTIMR2-$0212, VTIMR4-$0214). If two timer channels have been joined, place the address of the interrupt routine (LSB,MSB) in the vector register for the higher numbered channel, i.e., If channels number one and two have been joined, place the interrupt routine address in VTIMR2.

6. To enable the interrupts for the hardware timers, set the appropriate bit in IRQEN ($D20E), and the shadow register POKMSK ($0010). Setting bit 0 enables interrupts for timer 1, bit 1 enables interrupts for timer 2, and bit 2 enables interrupts for timer 4. Use an OR statement to accomplish this so the bits currently set in this register will not be altered (see example program).

7. To start the hardware timers, write any value to register STIMER ($D209).

HARDWARE TIMER LOCATIONS AND DEFINITIONS

AUDCTL - $D208(53768 decimal).

Bit	Result when set
0	Switches main clock base from 64 Khz to 15 Khz
3	Joins channel 4 to channel 3(16 bit resolution)
4	Joins channel 2 to channel 1(16 bit resolution)
5	Clocks channel 3 with 1.79 Mhz.
6	Clocks channel 1 with 1.79 Mhz.

IRQEN - $D20E(53774 decimal)
POKMSK - $0010(16 decimal) shadow register for IRQEN

Bit	Result when set
0	The POKEY timer #1 interrupt is enabled
1	The POKEY timer #2 interrupt is enabled
2	The POKEY timer #4 interrupt is enabled(OS rev B)

AUDIO CONTROL REGISTERS	Hex	Dec	AUDIO FREQ. REGISTERS	Hex	Dec
AUDC1	$D201	53761	AUDF1	$D200	53760
AUDC2	$D203	53763	AUDF2	$D202	53762
AUDC3	$D205	53765	AUDF3	$S204	53765
AUDC4	$D207	53767	AUDF4	$D206	53766

HARDWARE TIMER VECTOR LOCATIONS

Timer #	Timer Name	Hex Loc	Dec Loc
1	VTIMR1	$0210,$0211	528,529
2	VTIMR2	$0212,$0213	530,531
4	VTIMR4	$0214,$0215	532,533

STIMER - $D209 (53769) - Start hardware timers.
Store any non-zero value in STIMER to stat timers.

HARDWARE TIMER DEMONSTRATION PROGRAM
JC-5/10/83

The following demonstration program uses the hardware timer interrupt capability, to time the duration of each tone in a series of 256 tones. When all 256 tones have been played, the series is repeated. Timer number three is used to produce the tone, while a combination of timers one and two used to generate the interrupt that causes the next tone in the series to be played, by loading it into the AUDF3, audio register.

The use of hardware timers number one and two joined together, provide a demonstration of a wide range of timing values. The BASIC portion of the program allows a choice of clock speed, and number of cycles to count (duration). In selecting a clock speed of 15 Khz (15000 cycles per second), and choosing the maximum duration 65,535 cycles to count ($FFFF), it is possible to create a delay of 4,369 seconds between notes. Near the other end of the spectrum a choice of 1.79 Mhz in combination with a selection of 1791 cycles to count, will cause a delay of .0010 seconds between notes. At this rate, all 256 notes can be played in 0.26 seconds. The total number of cycles counted during this 0.26 second period is 458,496.

The purpose of this discussion is not to show how to generate sound effects, since this can be done without using the interrupt capability of the timer, but to point out the great speed that these timers are capable of and the large number of data samples that could be captured while monitoring an analog to digital converter connected to a real time control application.

To run the BASIC hardware timer demonstration program type in the program and enter RUN. When prompted to enter the timer duration, enter a decimal number from 1 to 65535. You will then be prompted to enter the clock rate to be used. Enter a 1 to use the 15 Khz rate, a 2 to use the 64 Khz rate or a 3 ti use the 1.79 Mhz rate. Try experimenting with different duration values and clock speeds. Remember that all 256 notes will be played regardless of the duration or clock speed chosen. To run the program again with different time and/or clock values, press System Reset key to halt program execution, and enter RUN; enter responses to prompts for duration and clock speed.

```
100 REM ****************************
110 REM *                          *
120 REM *     HARDTIME.BAS         *
130 REM *      JC 5/09/83          *
140 REM *                          *
150 REM * Hardware timer demo      *
160 REM * pokes machine language   *
170 REM * program into memory      *
180 REM * and calls with           *
190 REM * A=USR(1536)              *
200 REM *                          *
210 REM ****************************
220 REM
230 REM
240 ? CHR$(125):REM CLEAR SCREEN
250 FOR I=0 TO 84
260 READ J:POKE 1536+I,J
270 NEXT I
280 POSITION 2,10
290 REM ** Get timer duration **
300 ? "INPUT TIMER DURATION(0-65535)";:INPUT DUR
310 IF DUR<1 OR DUR>65535 THEN ? CHR$(253):GOTO 280
320 REM * Calculate MSB,LSB duration *
330 HIDUR=INT(DUR/256):LODUR=DUR-(HIDUR*256)
340 POSITION 2,12
350 ? "ENTER CLOCK SPEED":?
360 ? "          1= 15 Khz"
370 ? "          2= 64 Khz"
380 ? "          3= 1.79 Mhz":?
390 POSITION 2,20
400 ? "CLOCK SPEED = ";:INPUT CLOCK
410 IF CLOCK<1 OR CLOCK>3 THEN ? CHR$(253):GOTO 390
420 IF CLOCK=1 THEN CLOCK=17
430 IF CLOCK=2 THEN CLOCK=16
440 IF CLOCK=3 THEN CLOCK=80
450 POKE 54286,0:REM DISABLE VBI'S
460 POKE 54272,0:REM DISABLE DMA
470 POKE 1590,HIDUR:REM MSB DURATION
480 POKE 1585,LODUR:REM LSB DURATION
490 POKE 1572,CLOCK:REM SET CLOCK SPEED
500 A=USR(1536)
510 REM
520 REM
530 DATA 104,169,0,133,205,141,8,210,169,3,141,15,210,32,35,6,
        96,166,205,232,134,205,142,4,210,169,175,141,5,210
540 DATA 32,35,6,104,64,169,17,141,8,210,169,0,141,3,210,141,
        1,210,169,255,141,0,210,169,255,141,2,210,169,17
550 DATA 141,18,2,169,6,141,19,2,120,165,16,9,2,133,16,141,14,
        210,88,169,255,141,9,210,96
```

```
10 ;**********************************
20 ;* HARDWARE TIMER DEMONSTRATION  *
25 ;*                               *
30 ;* uses pokey hardware timer     *
35 ;* to time duration between notes*
40 ;* called from BASIC with       *
45 ;* A=USR(1536)
50 ;*                               *
55 ;**********************************
60 ;
65 AUDCTL = $D208   ;audio control
70 SKCTL = $D20F    ;serial port control
75 AUDC1 = $D201    ;audio channel 1 control
80 AUDC2 = $D203    ;audio channel 2 control
85 AUDC3 = $D205    ;audio channel 3 control
90 AUDF1 = $D200    ;audio frequency #1
95 AUDF2 = $D202    ;audio frequency #2
0100 AUDF3 = $D204  ;audio frequency #3
0105 VTIMR2 = $0212 ;pokey timer #2 vector
0110 STIMER = $D209 ;start pokey timers
0115 IRQEN = $D20E  ;interrupt request enable
0120 POKMSK = $010  ;IRQIN shadow
0125 NOTE = $CD     ;store note value here
0130 ;
0135 ;
0140  *= $600       ;place program on page six
0145  PLA           ;# arguments from BASIC
0150  LDA #0        ;initialized note to
0155  STA NOTE       ;save note value
0160  STA AUDCTL    ;initialize audio control
0165  LDA #$3       ;value to initialize POKEY
0170  STA SKCTL     ;initialize POKEY
0175  JSR SETUP
0180  RTS           ;return to BASIC
0185 ;
0190 ;
0195 ;**********************************
0200 ;*                               *
0205 ;*This routine will occur after*
0210 ;*each interrupt generated     *
0215 ;*by the hardware timer.       *
0220 ;*It will cause the next note  *
0225 ;*in the series to be played   *
0230 ;*and then initialize and      *
0235 ;*start the timer again       *
0240 ;*                               *
0245 ;**********************************
0250 ;
0255 TIME LDX NOTE ;get not from memory
0260  INX       ;increase to next note
0265  STX NOTE  ;store new note in memory
0270  STX AUDF3 ;place note in audio freq register
0275  LDA #$AF  ;use pure note full volume
0280  STA AUDC3 ;place in audio control register
0285  JSR SETUP ;initialize and start timer
```

```
0290  PLA
0295  RTI        ;return from interrupt
0300 ;
0305 ;
0310 ;*******************************
0315 ;*                             *
0320 ;*Routine to set and start     *
0325 ;*pokey hardware timer #2      *
0330 ;Clock is reduced to slowest   *
0335 ;*speed of 15 Khz.             *
0340 ;*value to timer is initialized*
0345 ;*at highest value $FFFF       *
0350 ;*(decimal 65535)              *
0355 ;*                            *
0360 ;*******************************
0365 ;
0370 ;
0375 SETUP LDA #$11   ;join channels two and 1
0380  STA AUDCTL      ;and switch clock base from 64 to 15 Khz
0385  LDA #0
0390  STA AUDC2       ;set timer channel
0395  STA AUDC1       ;audio volumes to 0
0400  LDA #$FF        ;LSB of timer duration
0405  STA AUDF1       ;audf1 holds LSB duration
0410  LDA #$FF        ;MSB of timer duration
0415  STA AUDF2       ;audf2 holds MSB  of duration
0420  LDA #TIME&255   ;LSB of address for timer interrupt
0425  S
0430  LDA TIME/256    ;MSB of address for timer interrupt
0435  STA VTIMR2+1    ;pokey timer #2 vector
0440  SEI             ;disable all maskable interrupts
0445  LDA POKMSK      ;value of interrupt enable state
0450  ORA #2          ;set bit #2 to enable pokey timer #2
0455  STA POKMSK      ;place this value in pokey shadow
0460  STA  IRQEN      ;also store in pokey hardware register
0465  CLI             ;enable interrupts
0470  LDA #$FF        ;use non zero value to start timers
0475  STA STIMER      ;start timer
0480  RTS             ;return to controlling routine
```

++

THE DREADED (or maybe infamous) ATARI KEYBOARD LOCK-UP

Here is a tip(?) from our friends at the Atari Computer Entusiasta (N.S.W) in Sidney Australia.

Lately, anumber of magazines have advised that there is a way out of the infamous Atari keyboard lock-up, apart from this I received a letter from FUTURETRONICS which mentioned the same method. I should mention that I have had absolutely NO luck with it but perhaps it will work in some situations. Let us know if you get it to work. (Hope you've got enough fingers')

Hold down the START, SELECT, and OPTION keys simultaneously. While holding these down, press and hold down the following keys in this order; SHIFT, CTRL, P, M and %. Then LIST the program and edit.

NEWSLETTER INFORMATION:

This newsletter is written and printed by members of the Milwaukee Area ATARI User's Group (MILATARI), an association of individuals with a common interest in using and programming ATARI computers. MILATARI is not affiliated with the ATARI company or any other commercial organizations.

All articles are written and donated by the membership. Opinions expressed in this publication are those of the individual author and do not necessarily represent, nor reflect, the opinions of MILATARI nor those of any other commercial or non-commercial organizations. Any article appearing in this newsletter may be reproduced, providing credit is given to the author and to MILATARI.

Your contribution of articles are always welcome. You may submit your article on ATARI compatible cassette, diskette, on typewritten form, or you can arrange with the editor to download your file via a modem at either 300 or 1200 BAUD. When submitting your article on cassette or diskette, please do not include any format control codes imbedded within the text. Deadline for articles is the last day of each month for inclusion on the next issue.

Write MILATARI NEWSLETTER, P.O. Box 1191, Waukesha, WI 53187-1191 for more information.

MEMBERSHIP INFORMATION:

Membership is open to individuals and families who are interested in using and programming ATARI computers. The membership includes a subscription to this newsletter and access to the clubs cassette, diskette and publication libraries.

There are 3 classes of memberships available. Associate, Individual and Family. Associate members can attend all club functions and may withdrawal materials from the club libraries. In addition to attending club functions and checking out materials from the libraries, Individual and Family members are entitled to vote in club elections and to hold elected position in the organization. The annual membership fees are $10.00 for associate, $15.00 for individual, and $20.00 for the family membership. Members are expect to abide by the by-laws of the club. You may receive a copy of the by-laws by contacting the club secretary.

For more information on how to join MILATARI, please contact the membership committee.

MEETING INFORMATION:

MILATARI meetings are held once monthly. The meetings are currently being held at the Armbruster School, 7000 Greenway, Greenfield. (Off 68th Street, behind Southridge Shopping Center.) The date of the meeting is the third Saturday of each month. Doors are open at 2:00PM. An agenda of the next meeting can be found elsewhere in this newsletter. For more specific details on the agenda for the next scheduled meeting, please contact the Vice President.

MILATARI Officers:

President	Gary Nolan	353-9716
Vice President	Chris Stieber	529-2663
Treasurer	David Frazer	542-7242
Secretary	open	
Education Committee	Linda Scott (Chairperson)	466-2314
	Joe Sanders (SIG Chairman)	447-1660
	Erick Hanson (Instructor)	252-3146
Cassette Librarian	Ron Friedel	354-1717
Disk Librarian	Bill Lawrence	968-3082
	Carl Mielcarek	355-3539
Publication Librarian	Marcus Hagen	445-0710
Membership Committee	Dennis J. Bogie	968-9341
	Sharon Gamache	421-2887
Newsletter Editor	David Frazer	542-7242
Bulletin Board SYSOP	Pete Kurth	355-6031(BBS)

* * * * * * * * * * *

TECHNICAL SUPPORT GROUP:

The following members have indicated a willingness to assist MILATARI members with programming and other related technical problems. Please be polite and do not call these members during meal periods or at very early or very late hours.

William Lawrence	Programming	1-968-3082
Don Wilcox	Programming	228-1650
Erik Hanson	Prog/Tech	252-3146
Steve Booth	Programming	367-8739
Nick Liberski	Prog/Tech	782-5594
David Frazer	Prog/Tech	542-7242

* * * * * * * * * *

MILATARI BULLETIN BOARD:

The Milwaukee Area ATARI Users Group maintains a 24 hour bulletin board service. This board is designed for the use of our members and other ATARI users around the country. The BBS allows for upload and downloading programs and files, a public message board and club news. The board operates at 300 BAUD. The phone number is (414)355-6031.

==========

MILATARI NEWSLETTER
P.O. Box 1191
Waukesha, Wisconsin 53187-1191

U.S. Postage
BULK RATE
PAID
Permit # 201
Waukesha, WI 53187

ADDRESS CORRECTION REQUESTED
FORWARDING POSTAGE GUARANTEED

==========